Bioinformática y análisis de datos ómicos

Tema 3: Manejo de Datos de Origen Biológico en R

Ignacio Varela Egocheaga

Departamento de Biología Molecular

Tema 3: Manejo de datos de origen biológico en R

3.1 Principios básicos de R: uso de RStudio

3.2 Almacenamiento de información en forma de vectores

3.3 Creacción y uso de matrices y cuadros de datos

3.4 Agrupación de distintos objetos en listas

3.5 Lectura y escritura de archivos

Tema 3: Manejo de datos de origen biológico en R

3.1 Principios básicos de R: uso de RStudio

3.2 Almacenamiento de información en forma de vectores

3.3 Creacción y uso de matrices y cuadros de datos

3.4 Agrupación de distintos objetos en listas

3.5 Lectura y escritura de archivos

Lenguajes de programación

1.1 Think algorithmically

By algorithmically, we mean in a precise *step-by-step* fashion where each action is equi
smallest possible increment. So, if you want a machine to make you a cup of tea, instea
`CoopR` ! Fetch me a cuppa!", you need to break everything down. Something like this:

1. `Go to kitchen`
2. `If no kitchen, jump to 53`
3. `Else get kettle`
4. `If no kettle, jump to 53`
5. `Check how much water in kettle`
6. `If is more or equal than 500 ml of water in kettle, jump to 15`
7. `Place kettle under cold tap`
8. `Open kettle lid`
9. `Turn cold tap on`
10. `Check how much water in kettle`
11. `If there is more than 500 ml of water in kettle, jump to 13`
12. `Jump to 10`
13. `Turn cold tap off`
14. `Close kettle lid`
15. `If kettle is on base, jump to 17`
16. `Place kettle on base`
17. `If base is plugged in mains, jump to 19`
18. `Plug base in mains`
19. `Turn kettle on`
20. `Get cup`
21. `If no cup, jump to 53`
22. `Get tea bag` *# yeah, we're not posh*
23. `If no tea bag, jump to 53`
24. `Put tea bag in cup`
25. `If kettle has boiled, jump to 27`
26. `Jump to 25`

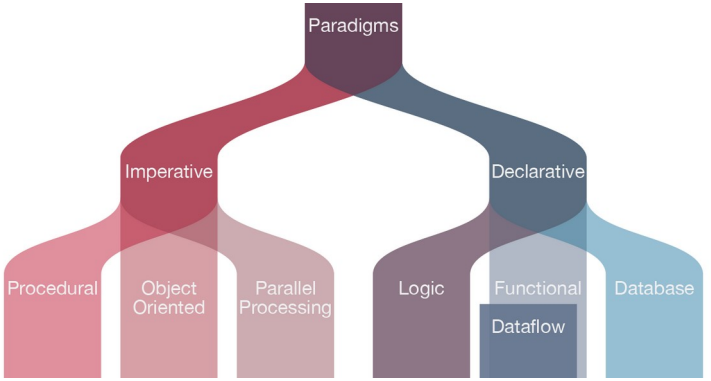
https://bookdown.org/animestina/R_Manchester/

Órdenes entendibles por el programador

Lenguaje programación

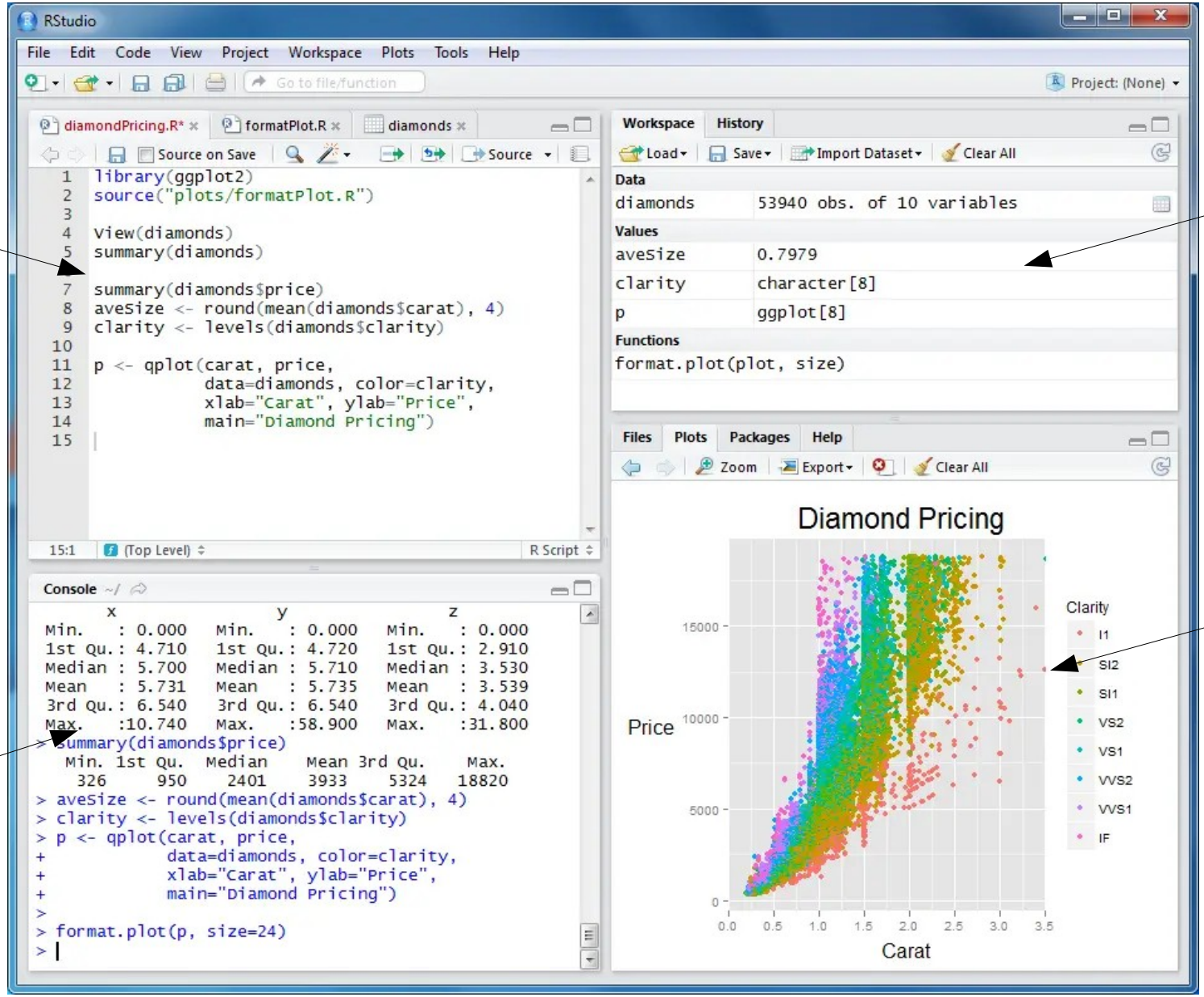
Intérprete

Órdenes entendibles por el ordenador



https://www.watelectronics.com/types-of-programming-languages-with-differences/

RStudio



Script/ programa

Variables/ Objetos

Gráficos y árbol de directorios

Ejecución de órdenes

Funciones

List of R Commands & Functions

- abline – Add straight lines to plot.
- abs – Compute the absolute value of a numeric data object.
- addmargins – Put margins on tables or arrays.
- addNA – Turn NA values into a factor level.
- aggregate – Compute summary statistics of subgroups of a data set.
- alist – Create a list object containing function arguments.
- all.equal – Test if two objects are nearly equal.
- all_equal [dplyr] – Compare two data frames.
- all_groups_identical [groupdata2] – Test if groupings of two factors are identical.
- all – Check whether all values of a logical vector are TRUE.
- anti_join [dplyr] – Anti join two data frames.
- any – Check whether any values of a logical vector are TRUE.
- apply – Apply function by rows or columns of a data frame.
- approx – Return a list of values that linearly interpolate given data points.
- approxfun – Return a function that performs linear interpolation.
- apropos – Return character vector with names of objects that contain the input.

...

...

- plogis – Return corresponding value of logistic CDF.
- plot – Draw a scatterplot or density plot.
- pmatch – Return position of first match between two data objects.
- pmax – Return the parallel maxima of two or more vectors.
- pmin – Return the parallel minima of two or more vectors.
- pnbinom – Return corresponding value of negative binomial cumulative distri
- pnorm – Return value of distribution function.
- polygon – Draw a polygon to a plot.
- ppois – Return value of poisson cumulative distribution function.
- pretty – Compute a sequence of equally spaced round values.
- print – Return data object to the R (or RStudio) console.
- prop.table – Create a proportions table.
- psignrank – Return corresponding CDF value of wilcoxon signedank statistic.
- pt – Return corresponding value of Student t CDF.
- ptukey – Return corresponding value of studentized range CDF.
- pull [dplyr] – Extract columns of data frame or tibble.
- punif – Return corresponding value of uniform CDF.
- pweibull – Return corresponding value of weibull CDF.
- pwilcox – Return corresponding CDF value of wilcoxon rank sum statistic.
- qbern – Return corresponding value of bernoulli quantile function.
- qbeta – Return corresponding value of beta quantile function.

...

```
help(print)
```

Description

`print` prints its argument and returns it *invisibly* (via `invisible(x)`). It is a generic function which means that new printing methods can be easily added for new `classes`.

Usage

```
print(x, ...)
```

Instalar nuevas funciones

Desde CRAN

```
install.packages("calendR")
library("calendR")
```

Desde Github

```
install.packages("devtools")
library("devtools")
install_github("tidyverse/ggplot2")
```

Desde Bioconductor

```
install.packages("BiocManager")
BiocManager::install("nanotatoR")
```

Almacenamiento en variables

Asignación de un nombre a un valor (← o =)

```
print("hello")

message ← "hello"
print(message)
```

Normas de los nombres de variables

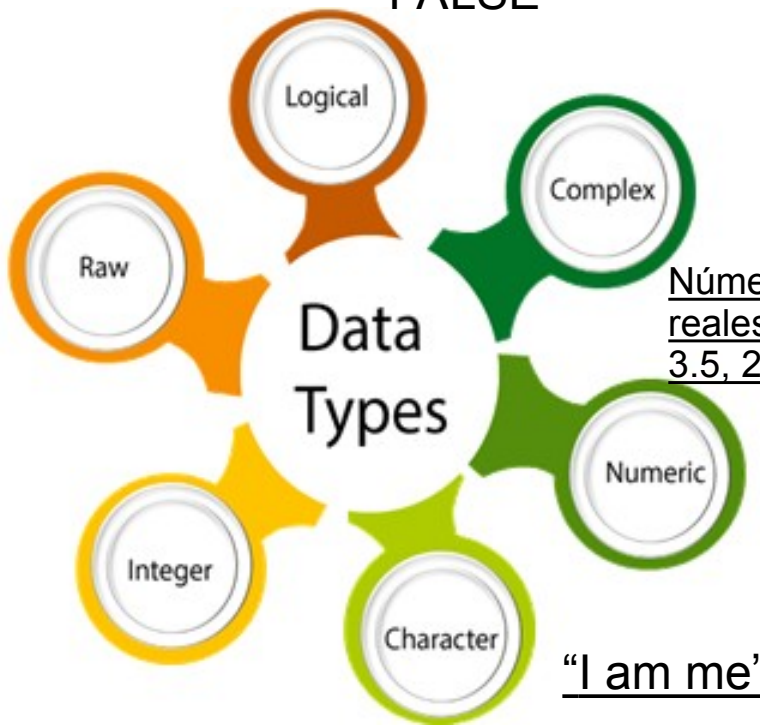
1.- Puede contener letras, números, punto o guión bajo
2.- Debe empezar por letra.
3.- Hay palabras reservadas por el lenguaje.

Operaciones con variables

```
a ← 5
b ← 3

a + b
a – b

c ← a + b

print(c)
```

Operador aritmético en R	Descripción
+	Suma
–	Resta
*	Multiplicación
/	División
^	Exponencial
**	Exponencial
%%	Módulo
%/%	División entera
%*%	Multiplicación matricial
%o%	Producto exterior
%x%	Producto Kronecker

TRUE/
FALSE



Números reales: 1, 3.5, 2.86,...

"I am me"

https://www.javatpoint.com/r-data-types

Operador lógico en R	Descripción
&	Comparación lógica 'AND' (Y) elemento a elemento
&&	Comparación lógica 'AND' de vectores
|	Comparación lógica 'OR' (O) elemento a elemento
||	Comparación lógica 'OR' de vectores
!	Negación lógica 'NOT' (NO)
xor()	Exclusión 'OR' elemento a elemento equivalente a !(x | y)

https://r-coder.com/inicio/

Tema 3: Manejo de datos de origen biológico en R

3.1 Principios básicos de R: uso de RStudio

3.2 Almacenamiento de información en forma de vectores

3.3 Creacción y uso de matrices y cuadros de datos

3.4 Agrupación de distintos objetos en listas

3.5 Lectura y escritura de archivos

Vectores

c → concatenar distintas variables **del mismo tipo** en un solo objeto.

```
Saludos ← c("Hello", "Hola", "Salut")
Alturas ← c(1.95, 2.27, 1.63, 1.54)
Obesidad ← c(TRUE, FALSE, FALSE, TRUE, TRUE)
```

Acceder a los elementos de un vector por índice o por nombre

1 2 3
Saludos ← c("Hello", "Hola", "Salut")

Saludos[1] → "Hello".

Saludos[2] → "Hola".

Saludos[3] → "Salut".

```
setNames(Saludos, c("Inglés", "Español", "Francés"))
```

ó

```
Saludos ← c("Inglés"="Hello", "Español"= "Hola", "Francés"="Salut")
```

Inglés Español Francés
Saludos ← c("Hello", "Hola", "Salut")

Saludos["Inglés"] → "Hello".

Saludos["Español"] → "Hola".

Saludos["Francés"] → "Salut".

Operaciones con vectores

El resultado de cualquier operación sobre un vector es otro vector

Ordenar un vector

```
Alturas_ordenadas ← sort(Alturas)   #creciente
Alturas_ordenadas_dec ← sort(Alturas, decreasing = TRUE)  #decreciente
```

Operaciones aritméticas (resultado = vector numérico)

```
X ← (5, 7, 8)
Y ← (2, 4, 9)
Z ← X + Y                    # Z = (7,11,17).
```

Operaciones lógicas (resultado = vector lógico)

```
X ← (5, 7, 8)
Y ← (2, 4, 9)
Z ← X > Y                    # Z = (TRUE, TRUE, FALSE).
```

Concatenar vectores

```
X ← (5, 7, 8)
Y ← (2, 4, 9)
Z ← c(X,Y)                    # Z = (5,7,8,2,4,9).
```

Aplicar una misma función a todos los elementos de un vector

```
X ← (5, 7, 8)
Z ← sapply(X,sqrt)                    # Z = (2.236068, 2.645751, 2.828427)
```

Generación automática de vectores numéricos

Vectores secuenciales

Utilizando expresión abreviada **inicio:final**

```
z ← 1:4      #z = (1,2,3,4)
```

Utilizando función **seq** seq(from,to,[by])

```
z ← seq(1,4)       #z = (1,2,3,4)
z ← seq(1,4,0.5)  #z = (1, 1.5, 2, 2.5, 3, 3.5, 4)
```

Vectores repetitivos

```
z ← rep(1,4)      #z = (1,1,1,1)
```

Vectores aleatorios

```
z ← sample(1:4, size=5, replace=TRUE)      #z = (1,3,2,2,4)
z ← rnorm(5, mean=0, sd= 1)    #z = (-1.56, -0.25,  -1.99, 0.30, -1.48)
```

Factores

```
z ← as.factor(c("alto","medio","medio","alto","alto","bajo","medio","bajo"))
```

Acceso a elementos de un vector usando otro vector

Usando vectores de índices

```
meses ← c("Enero","Febrero","Marzo","Abril","Mayo","Junio","Agosto")
índices ← c(1,4,7)  # índices = (1, 4, 7)
z ← meses[Índices] # z = ("Enero","Abril","Agosto")
```

ó

```
z ← meses[c(1,4,7)]  # z = ("Enero","Abril","Agosto")
```

```
z ← meses[seq(1,3)]  # z = ("Enero","Febrero","Marzo")
```

```
z ← meses[rep(1,3)]  # z = ("Enero","Enero","Enero")
```

Usando vectores lógicos

```
meses ← c("Enero","Febrero","Marzo","Abril","Mayo","Junio","Agosto")
días ← c(31, 28, 31, 30, 31, 30, 31)
mayores_30 ← días > 30  #mayores_30 = (T, F, T, F, T, F, T)
z ← meses[mayores_30] # z = ("Enero", "Marzo", "Mayo", "Agosto")
```

ó

```
z ← meses[días>30] # z = ("Enero", "Marzo", "Mayo", "Agosto")
```

Tema 3: Manejo de datos de origen biológico en R

3.1 Principios básicos de R: uso de RStudio

3.2 Almacenamiento de información en forma de vectores

3.3 Creacción y uso de matrices y cuadros de datos

3.4 Agrupación de distintos objetos en listas

3.5 Lectura y escritura de archivos

Matrices

matrix → agrupar de manera bidimensional variables **del mismo tipo** en un solo objeto.

```
numeros ← matrix(1:6, ncol = 2)
```

```
     [,1] [,2]
[1,]    1    4
[2,]    2    5
[3,]    3    6
```

numeros[3,2] → 6

```
numeros2 ← matrix(1:6, ncol = 2, byrow = TRUE)
```

```
     [,1] [,2]
[1,]    1    2
[2,]    3    4
[3,]    5    6
```

Combinando vectores o matrices

```
vector1 ← c("a","b","c)
vector2 ← c("d","e","f")
```

```
matriz1 ← rbind(vector1,vector2)
```

```
        [,1] [,2] [,3]
vector1    a    b    c
vector2    d    e    f
```

```
matriz2 ← cbind(vector1,vector2)
```

```
     vector1 vector2
[1,]       a       d
[2,]       b       e
[3,]       c       f
```

```
matriz3 ← cbind(numeros,numeros2)
```

```
     [,1] [,2] [,3] [,4]
[1,]    1    4    1    2
[2,]    2    5    3    4
[3,]    3    6    5    6
```

```
colnames(matriz3) ← c("Col1","Col2","Col3","Col4")
rownames(matriz3) ← c("Fila1","Fila2","Fila3")
```

```
      Col1 Col2 Col3 Col4
Fila1    1    4    1    2
Fila2    2    5    3    4
Fila3    3    6    5    6
```

matriz3["Fila3","Col2"] → 6

Operaciones sobre matrices

Operaciones aritméticas

```
numeros3 ← numeros + numeros2
```

```
     [,1] [,2]
[1,]    1    4
[2,]    2    5
[3,]    3    6
```

+

```
     [,1] [,2]
[1,]    1    2
[2,]    3    4
[3,]    5    6
```

→

```
     [,1] [,2]
[1,]    2    6
[2,]    5    9
[3,]    8   12
```

Transponer

```
numeros4 ← t(numeros)
```

```
     [,1] [,2]
[1,]    1    4
[2,]    2    5
[3,]    3    6
```

→

```
     [,1] [,2] [,3]
[1,]    1    2    3
[2,]    4    5    6
```

Operaciones sobre filas o columnas enteras

```
numeros5 ← apply(numeros,1,sum) # (1 → filas)
numeros6 ← apply(numeros,2,sum) # (2 → columnas)
```

numeros5 → (5, 7, 9)
numeros6 → (6, 15)

Acceso a elementos de matrices usando vectores

Vectores numéricos de índices

numeros

```
     [,1] [,2]
[1,]    1    4
[2,]    2    5
[3,]    3    6
```

```
elementos ← c(1,3)  #  (1,3)
seleccion ← numeros[elementos,]
```

ó

```
seleccion ← numeros[c(1,3),]
```

seleccion

```
     [,1] [,2]
[1,]    1    4
[2,]    3    6
```

Vectores lógicos

```
elementos ← numeros[ ,2] > 4    # (FALSE, TRUE, TRUE)
seleccion2 ← numeros[elementos,]
```

ó

```
seleccion2 ← numeros[numeros[,2] > 4,]
```

seleccion2

```
     [,1] [,2]
[1,]    2    5
[2,]    3    6
```

Cuadros de datos o *dataframes*

data.frame → agrupar de manera bidimensional variables **de distinto tipo** en un solo objeto.

```
alumnos ← c(“Paco”,”María”,”Julia”,”Lorena”)
alturas ← c(1.78, 1.68, 1.90, 1.85)
pesos ← c( 80, 57, 70, 65)
clase ← data.frame(nombre = alumnos, altura = alturas, peso = pesos)
```

	nombre	**altura**	**peso**
1	Paco	1.78	80
2	María	1.68	57
3	Julia	1.90	70
4	Lorena	1.85	65

```
rownames(clase) <- c("Alumno1","Alumno2","Alumno3","Alumno4")
```

	nombre	**altura**	**peso**
Alumno1	Paco	1.78	80
Alumno2	María	1.68	57
Alumno3	Julia	1.90	70
Alumno4	Lorena	1.85	65

Acceso a elementos del cuadro de datos

Clase[2,3] → 57 Clase[“Alumno4”,”altura”] → 1.85

Operaciones sobre cuadros de datos

Generación de nuevas variables operando sobre variables

	nombre	altura	peso
Alumno1	Paco	1.78	80
Alumno2	María	1.68	57
Alumno3	Julia	1.90	70
Alumno4	Lorena	1.85	65

```
clase$IMC <- clase$peso / (clase$altura)^2
```

	nombre	altura	peso	IMC
Alumno1	Paco	1.78	80	25.25
Alumno2	María	1.68	57	20.19
Alumno3	Julia	1.90	70	19.39
Alumno4	Lorena	1.85	65	18.99

Fusionar cuadros de datos

```
Aula ← c("aula3", "aula2","aula1","aula2")
clase2 ← cbind(clase,Aula)
```

	nombre	altura	peso	IMC	Aula
Alumno1	Paco	1.78	80	25.25	aula3
Alumno2	María	1.68	57	20.19	aula2
Alumno3	Julia	1.90	70	19.39	aula1
Alumno4	Lorena	1.85	65	18.99	aula2

Filtrando elementos del cuadro de datos

	nombre	altura	peso	IMC	Aula
Alumno1	Paco	1.78	80	25.25	aula3
Alumno2	María	1.68	57	20.19	aula2
Alumno3	Julia	1.90	70	19.39	aula1
Alumno4	Lorena	1.85	65	18.99	aula2

Vectores numéricos de índices

```
clase2[1:3,c(3,5)]
```

	peso	Aula
Alumno1	80	aula3
Alumno2	57	aula2
Alumno3	70	aula1

Vectores lógicos

```
clase2[clase2$peso>60,]
```

	nombre	altura	peso	IMC	Aula
Alumno1	Paco	1.78	80	25.25	aula3
Alumno3	Julia	1.90	70	19.39	aula1
Alumno4	Lorena	1.85	65	18.99	aula2

Tema 3: Manejo de datos de origen biológico en R

3.1 Principios básicos de R: uso de RStudio

3.2 Almacenamiento de información en forma de vectores

3.3 Creacción y uso de matrices y cuadros de datos

3.4 Agrupación de distintos objetos en listas

3.5 Lectura y escritura de archivos

Listas

Las listas permiten agrupar objetos de distintos tipos

```
lista ← list(pesos,clase2,numeros)
```

```
[[1]]
[1] 80 57 70 65
```

#vector

```
[[2]]
        nombre altura peso   IMC  Aula
Alumno1   Paco   1.78   80 25.25 aula3
Alumno2  María   1.68   57 20.20 aula2
Alumno3  Julia   1.90   70 19.40 aula1
Alumno4 Lorena   1.85   65 18.99 aula2
```

#dataframe

```
[[3]]
     [,1] [,2]
[1,]    1    4
[2,]    2    5
[3,]    3    6
```

#matriz

lista[[3]] → matriz

```
names(lista) ← c("Pesos","Clase","notas")
```

```
"Pesos"
[1] 80 57 70 65

"Clase"
        nombre altura peso   IMC  Aula
Alumno1   Paco   1.78   80 25.25 aula3
Alumno2  María   1.68   57 20.20 aula2
Alumno3  Julia   1.90   70 19.40 aula1
Alumno4 Lorena   1.85   65 18.99 aula2
```

```
"notas"
     [,1] [,2]
[1,]    1    4
[2,]    2    5
[3,]    3    6
```

lista$Clase → dataframe

Tema 3: Manejo de datos de origen biológico en R

3.1 Principios básicos de R: uso de RStudio

3.2 Almacenamiento de información en forma de vectores

3.3 Creacción y uso de matrices y cuadros de datos

3.4 Agrupación de distintos objetos en listas

3.5 Lectura y escritura de archivos

Salvar y recuperar objetos de R

Los objetos se pueden guardar y recuperar encriptados en un código binario particular. Es común utilizar la extensión **.RData** para estos archivos.

save(object(s) , file = ...)

```
save(clase2, file = "./clase2.RData")
```

En cualquier momento, estos objetos se pueden volver a cargar en el programa en cualquier otro momento.

load(file = ...)

```
load( file = "./clase2.RData")
```

También se puede grabar todos los objetos creados de una sola vez.

save.image(file = ...)

```
Save.image(file = "./session.RData")
```

Leer y escribir archivos

La mayoría de las funciones de lectura de archivos de texto generan un objeto del tipo cuadro de datos

read.delim2(file = …, header = TRUE, sep =”\t”, dec = “.” …)

```
file_dataframe ← read.delim2(file = “../dataframe.csv”, header=TRUE, sep = “,”)
```

Otras variantes de esta función: **read.table** , **read.csv**, **read.csv2**, **read.delim** o **read.xls**

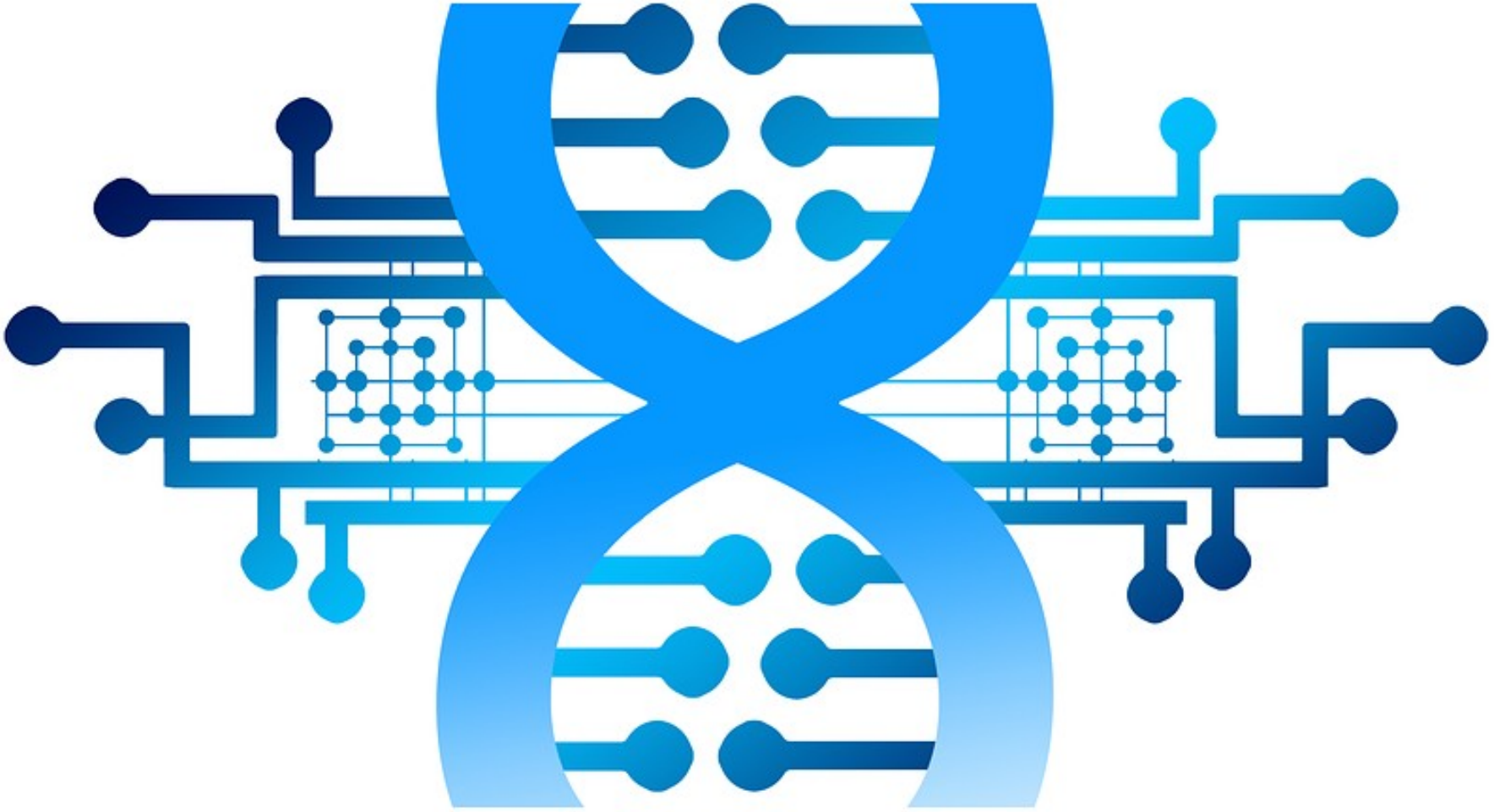
De manera similar, para grabar objetos en archivos de texto, usamos la función genérica write.table

write.table(object(s), file = …,sep =” ”, dec = “.”, row.names = TRUE, col.names = TRUE, …)

```
write.table(file_dataframe, file = “../dataframe2.csv”)
```

Otras variantes de esta función: **read.table** , **write.csv**, **write.csv2** o **write.xls**

Bioinformática y análisis de datos ómicos



Tema 3: Manejo de datos de origen biológico en R

ignacio.varela@unican.es